IoT for Smart Grids

FRANCESCO PAGANO I ERDEM AGFIRAT

Contents

Overview

INCREASING INTEGRATION OF RENEWABLE ENERGIES, ESPECIALLY IN LOW-VOLTAGE GRIDS

GROWING ELECTRICAL ENERGY DEMAND DUE TO ELECTRIFICATION OF THE MOBILITY AND HEATING SECTORS

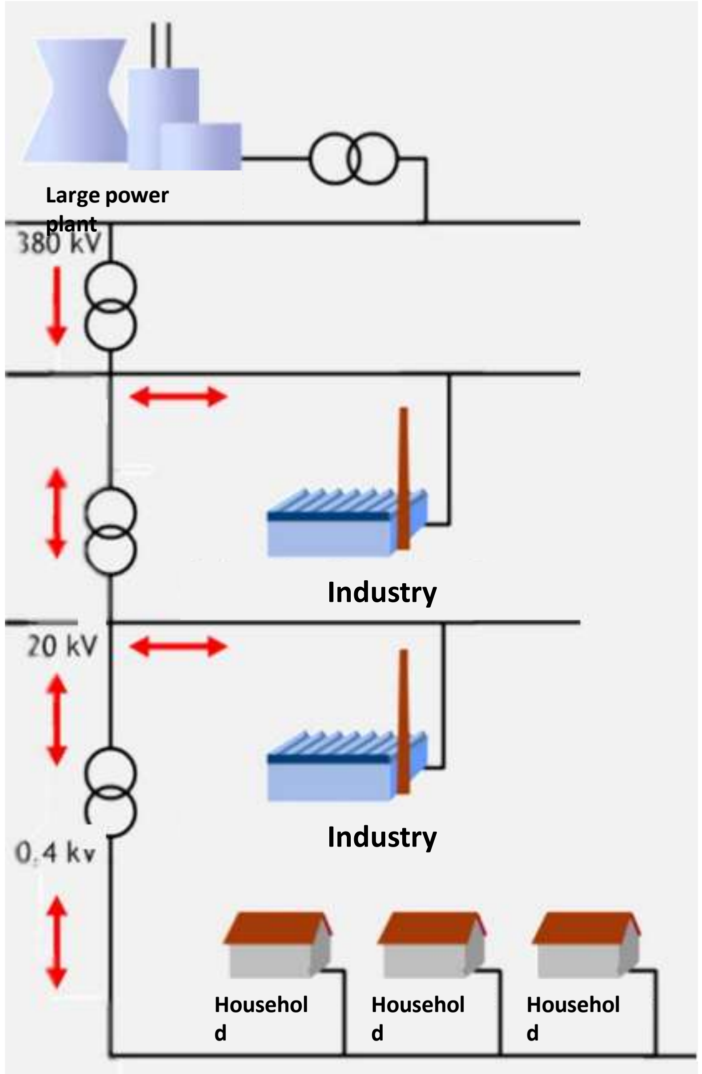
RESULTING IN NEW STRESS CONDITIONS ON THE GRID

INFORMATION AND COMMUNICATION TECHNOLOGIES (ICT) WILL PLAY A CENTRAL ROLE IN CONNECTING COMPONENTS OF THE ENERGY SYSTEMS

SMART GRIDS ENABLE REAL-TIME COMMUNICATION AND COORDINATION BETWEEN VARIOUS ENERGY SYSTEM ACTORS

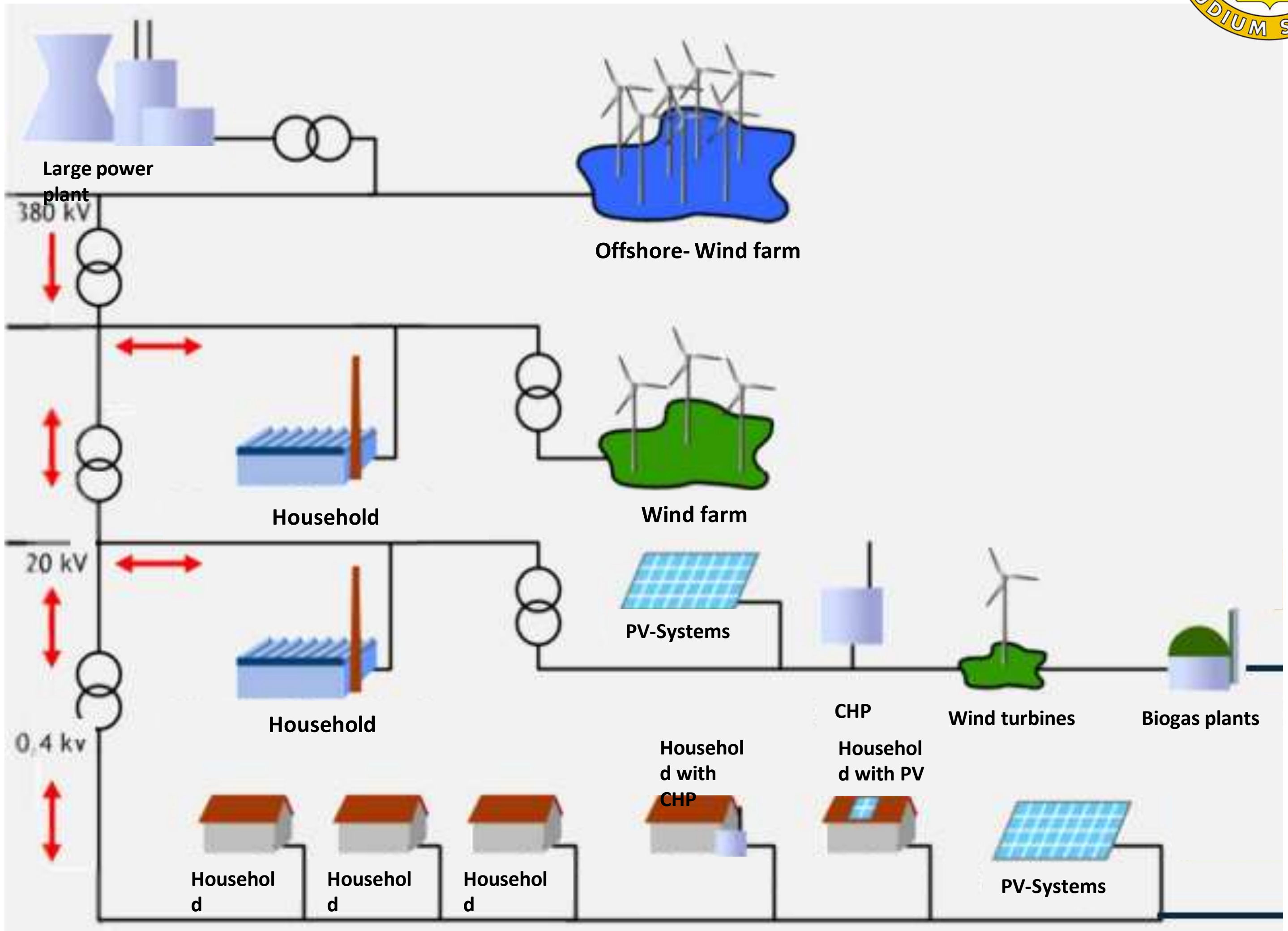
Yesterday

- Top-down approach
- Large power plants
- Consumers have little flexibility
- Unidirectional power flow



Today and Tomorrow

- Many smaller, decentralized generation units
- Generation in the distribution grid
- Bidirectional power flow
- Electrification of new sectors
  - Increasing electricity consumption
- Consequence: Deployment of flexibilities



The goal of our project

- To simulate a realistic future scenario:
  - Two types of houses have been designed: single-family houses and skyscrapers
  - Some houses are equipped with solar panels, some with batteries, and others with electric cars
  - One objective is to display the current and voltage in real-time, simulating a smart meter
  - Another objective is to store excess electricity generated by houses and distributed it to other houses as required (e.g., for electric cars)
  - These features contribute to the intelligence of the network

Houses

Houses are made of wood

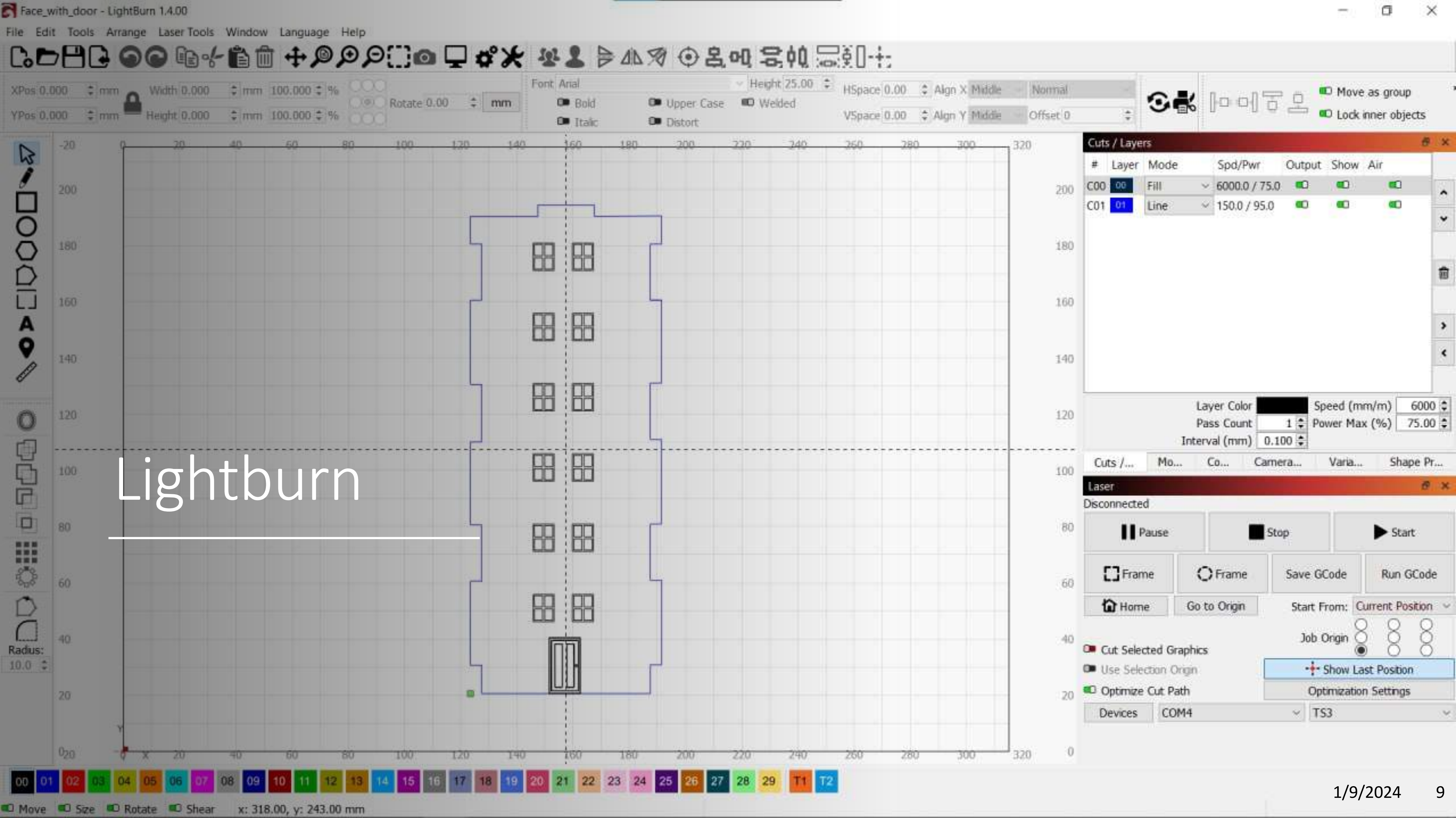
Used Laser and Software:

- Twotrees TS3 Enclosed Professional 4th Axis Laser Engraver and Cutter Machine
- Lightburn

LIGHTBURN

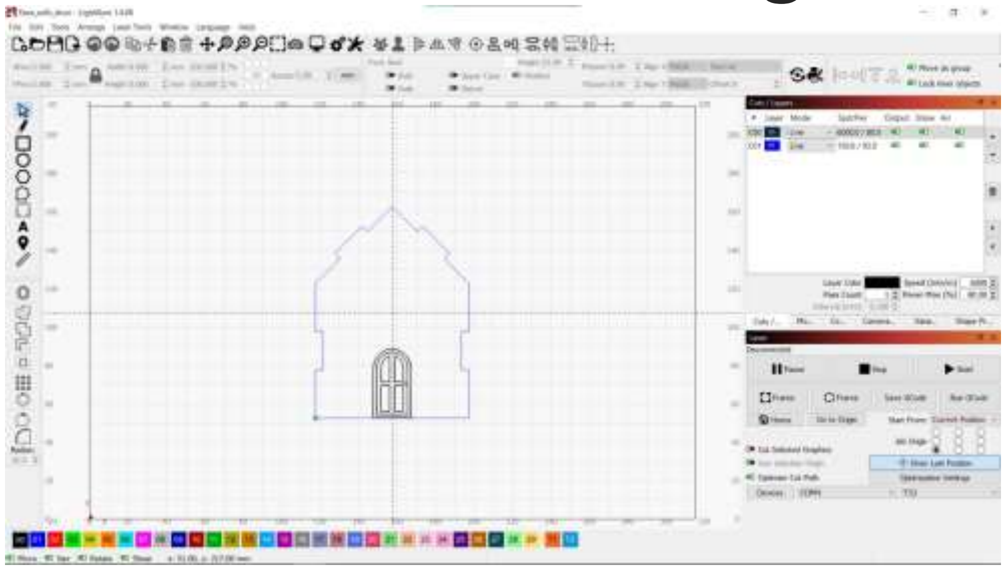
BETTER SOFTWARE FOR LASER CUTTERS

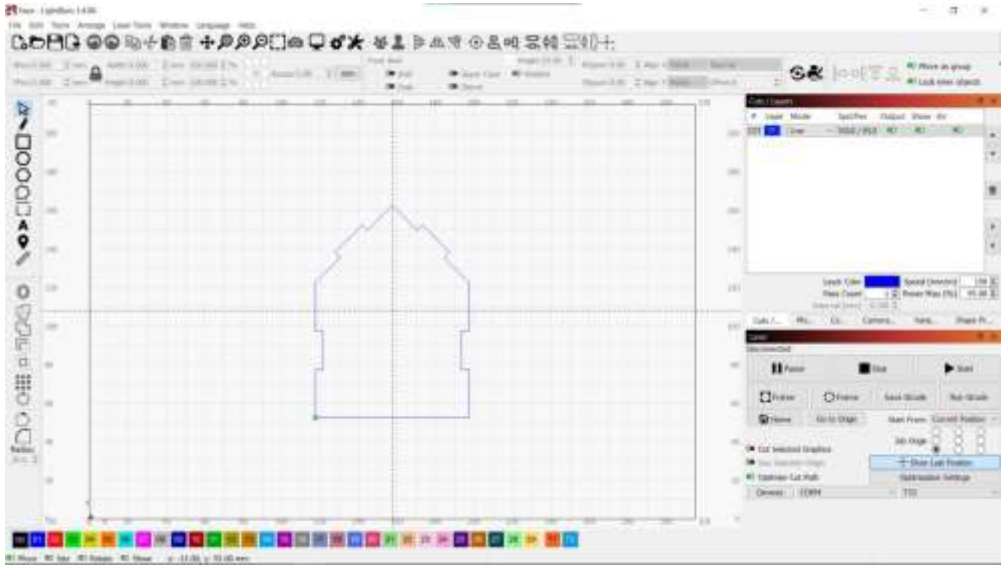
Lightburn

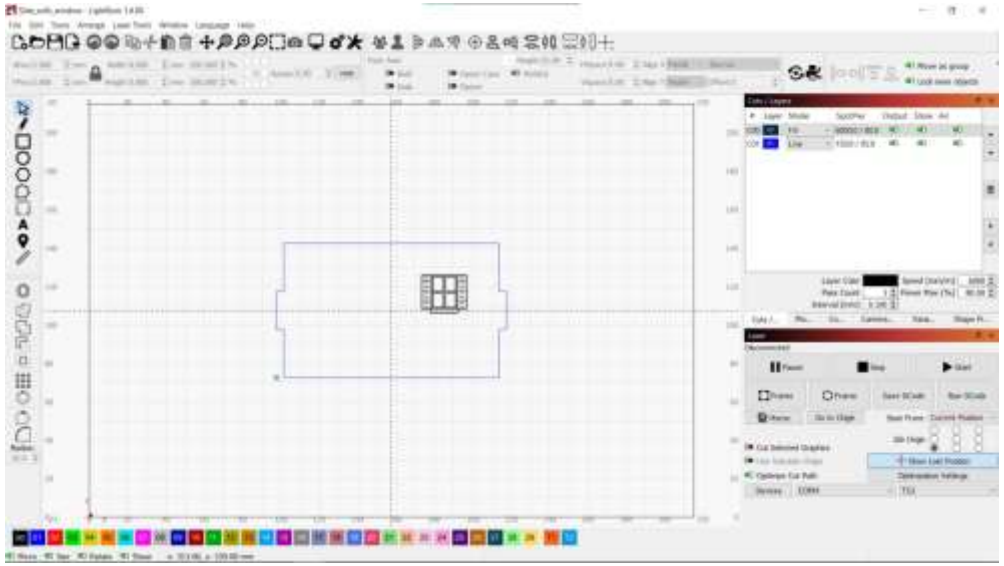


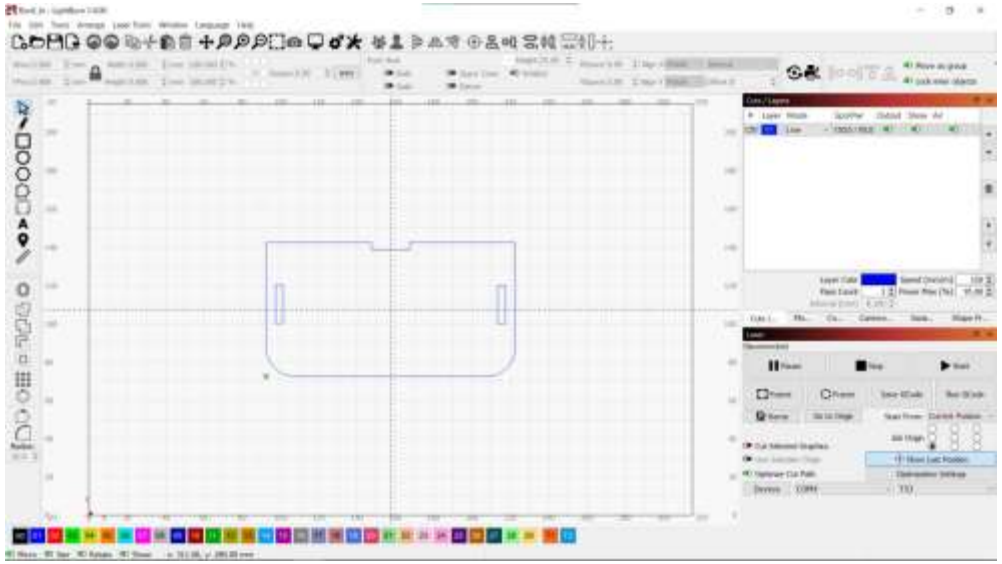
Face_with_door - LightBurn 1.4.00
File Edit Tools Arrange Laser Tools Window Language Help
XPos 0.000 mm
YPos 0.000 mm
Width 0.000 mm 100.000 %
Height 0.000 mm 100.000 %
Rotate 0.00
mm
Font Arial
Height 25.00
Bold
Italic
Upper Case
Distort
Welded
HSpace 0.00
VSpace 0.00
Align X Middle
Align Y Middle
Normal
Offset 0
Move as group
Lock inner objects
Cuts / Layers
Layer Mode Spd/Pwr Output Show Air
C00 00 Fill 6000.0 / 75.0
C01 01 Line 150.0 / 95.0
Layer Color
Speed (mm/m) 6000
Pass Count 1
Power Max (%) 75.00
Interval (mm) 0.100
Cuts /... Mo... Co... Camera... Varia... Shape Pr...
Laser
Disconnected
Pause
Stop
Start
Frame
Frame
Save GCode
Run GCode
Home
Go to Origin
Start From: Current Position
Job Origin
Cut Selected Graphics
Use Selection Origin
Show Last Position
Optimize Cut Path
Optimization Settings
Devices
COM4
TS3
Radius:
10.0
Move Size Rotate Shear x: 318.00, y: 243.00 mm

Lightburn – Houses









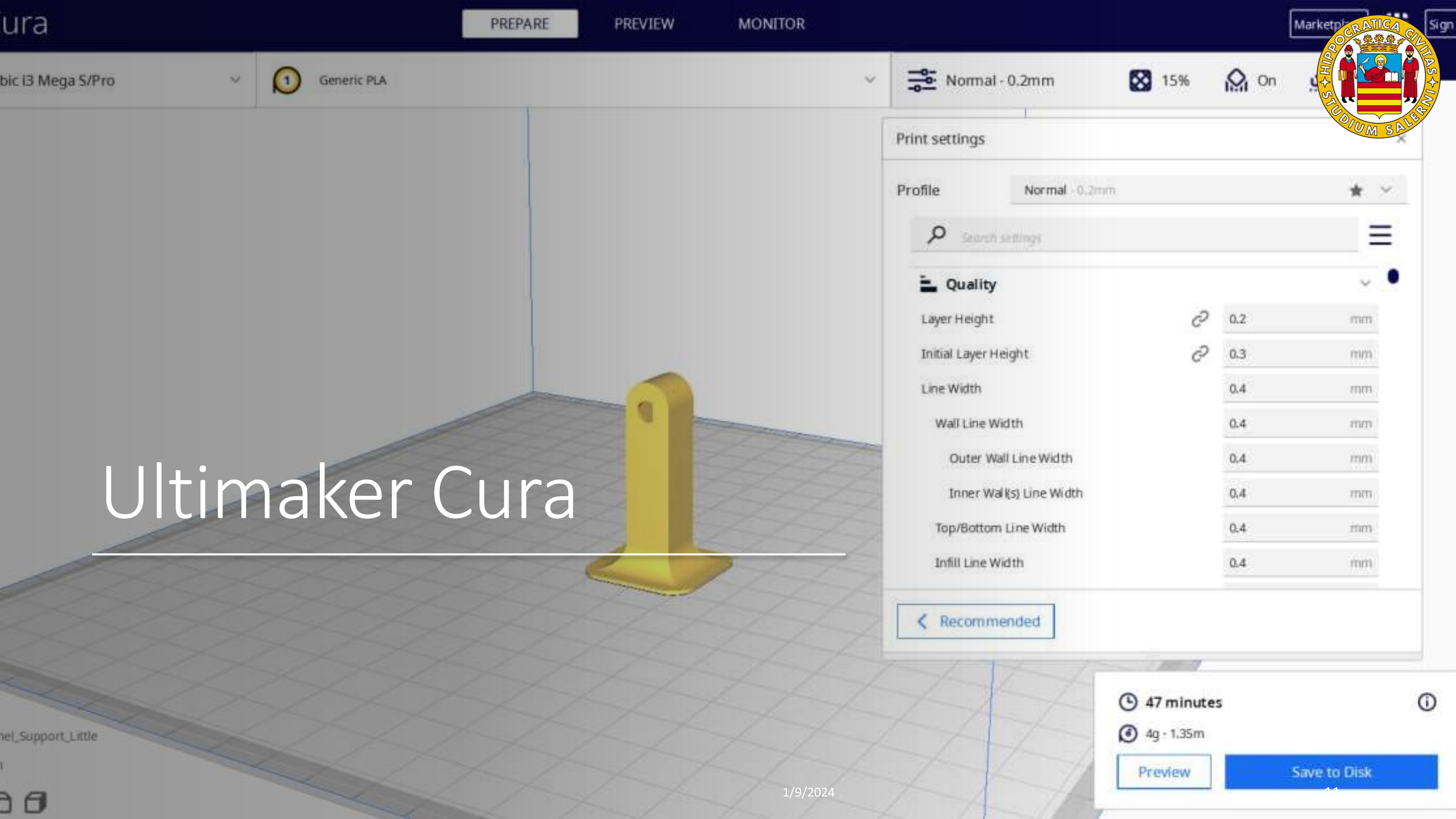
Details for the Project – 3D Printer

The supports for the micro solarpanels are made of the material PETG

The Angel of the support is 45° to present the most realistic scenario possible

The used devices and Software:

- Anycubic i3 Mega S
- UltiMaker Cura



Ultimaker Cura

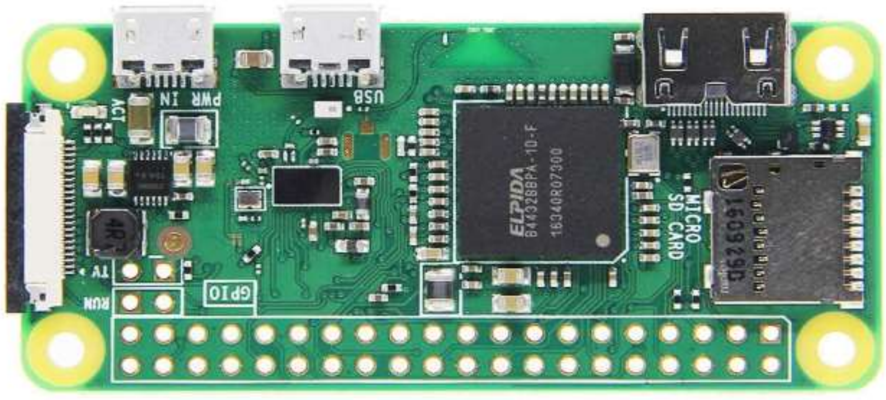
IoT for Smart Grids – HELTEC – LoRa

- Range: 5 – 20 Km
- Date rate: 37,5 Kbps
- Operating area: 867 – 869 MHz
- Ideal for IoT because it allows long range with low power consumption and secure data transmission

HELTEC LoRa 32 (v2)

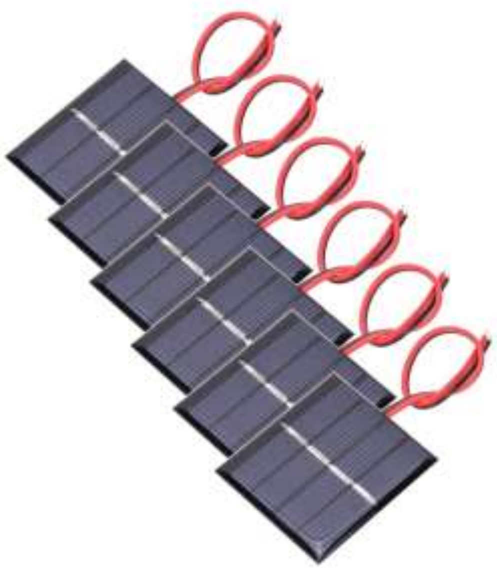
IoT for Smart Grids – Raspberry Pi Zero

- To control the relays
- RPC represents a form of client-server interaction
- Remote procedure call = a program causes a procedure to execute in a different address space



Raspberry Pi Zero

IoT for Smart Grids – Hardware



Solar panels

Relay

IoT for Smart Grids – Hardware

- RGB and white light detection
- Output RGB data and light intensity through the **I2C**
- Its advantages include high sensitivity, wide dynamic range, accurate measuring

TCS34725

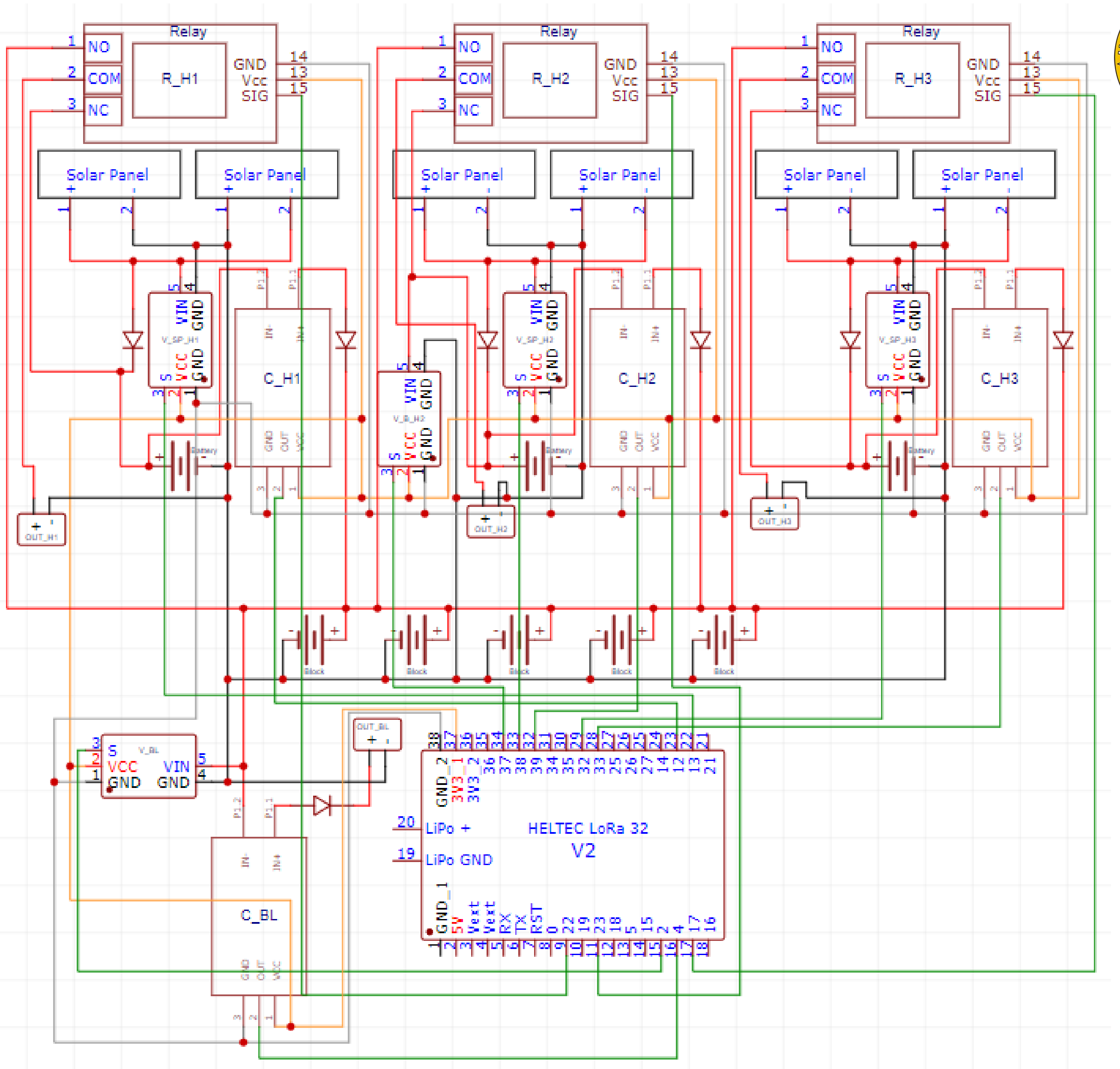
Current and voltage sensors

- Current and voltage sensors are crucial in our project
- They read the power generated by the solar panels
- They can simulate a Smart Meter, which play a important role in Smart Grids
- The collected data enables informed decisions, such as determining if energy is needed from the load

- Voltage Sensor

- Current Sensor

Schematic


Relay
NO
COM
NC
R_H1
R_H2
R_H3
GND
Vcc
SIG
Solar Panel
C_H1
C_H2
C_H3
VIN
GND
VCC
S
OUT_H1
OUT_H2
OUT_H3
OUT_BL
V_BL
C_BL
Battery
Block
HELTEC LoRa 32
V2
LiPo +
LiPo GND
GND_2
3V3_1
3V3_2
GND_1
5V
Vext
RX
TX
RST

SLAVE

```
void loop() {
  analogReadResolution(12);

  float voltP_H2 = read_voltage(VOLTAGE_PANEL_H2, offset_voltage);
  float voltP_H4 = read_voltage(VOLTAGE_PANEL_H4, offset_voltage);
  float voltB_H4 = read_voltage(VOLTAGE_BATTERY_H4,  offset_voltage);

  float cur_H2 = read_current(CURRENT_H2);
  float cur_H4 = read_current(CURRENT_H4);
  float cur_H6 = read_current(CURRENT_H6);

  unsigned char buffer[255];
  delay(5000);

  //HOUSE_2
  int index = 0;
  buffer[index++] = 0;
  insertIntoBuffer(&index, buffer, voltP_H2);
  insertIntoBuffer(&index, buffer, cur_H2);
  sendBuffer(buffer, index);
  delay(3000);

  //HOUSE_4
  index = 0;
  buffer[index++] = 1;
  insertIntoBuffer(&index, buffer, voltP_H4);
  insertIntoBuffer(&index, buffer, voltB_H4);
  insertIntoBuffer(&index, buffer, cur_H4);
  sendBuffer(buffer, index);
  delay(5000);

  //HOUSE_6
  index = 0;
  buffer[index++] = 2;
  insertIntoBuffer(&index, buffer, cur_H6);
  sendBuffer(buffer, index);
  delay(1000);
}

float read_voltage(int PIN, float offset){
  int adc_value = analogRead(PIN);
  float adc_voltage  = (adc_value * ref_voltage) / 4095.0;
  float in_voltage = adc_voltage / (R2/(R1+R2)) + offset;
```

MASTER

```
if (rssi != 0) {
  byte sensorType = buffer[0];

  float voltP_H2, cur_H2, voltP_H4, voltB_H4, cur_H4, cur_H6;

  StaticJsonDocument<1024> jsonDoc;
  String jsonDoc_string;

  switch (sensorType){
    case 0:
      jsonDoc.clear();
      readFloatFromBuffer(buffer, 1, &voltP_H2);
      readFloatFromBuffer(buffer, 5, &cur_H2);
      jsonDoc["Voltage_Panel_H2"] = voltP_H2;
      jsonDoc["Current_H2"] = cur_H2;
      serializeJson(jsonDoc, jsonDoc_string);
      sendTelemetry(jsonDoc_string.c_str(), H2_TOKEN);
      Serial.println(jsonDoc_string);
      break;
      //delay(2000);
    case 1:
      jsonDoc.clear();
      readFloatFromBuffer(buffer, 1, &voltP_H4);
      readFloatFromBuffer(buffer, 5, &voltB_H4);
      readFloatFromBuffer(buffer, 9, &cur_H4);
      jsonDoc["Voltage_Panel_H4"] = voltP_H4;
      jsonDoc["Voltage_Battery_H4"] = voltB_H4;
      jsonDoc["Current_H4"] = cur_H4;
      serializeJson(jsonDoc, jsonDoc_string);
      sendTelemetry(jsonDoc_string.c_str(), H4_TOKEN);
      Serial.println(jsonDoc_string);
      break;
    case 2:
      jsonDoc.clear();
      readFloatFromBuffer(buffer, 1, &cur_H6);
      jsonDoc["Current_H6"] = cur_H6;
      serializeJson(jsonDoc, jsonDoc_string);
      sendTelemetry(jsonDoc_string.c_str(), H6_TOKEN);
      Serial.println(jsonDoc_string);
      break;
      delay(1000);
      }
```

Raspberry Code (RPC - Actuators)

```python
#!/usr/bin/python3
import os, time, sys, json, random
import paho.mqtt.client as mqtt
import RPi.GPIO as GPIO
import metodi, keys

def on_connect(client, userdata, flags, rc):
        client.subscribe('v1/devices/me/rpc/request/+')

def on_message(client, userdata, msg):
        if msg.topic.startswith('v1/devices/me/rpc/request/'):
                requestId = msg.topic[len('v1/devices/me/rpc/request/'):len(msg.topic)]
                data = json.loads(msg.payload)

        if data['method'] == 'accendiH1':
                metodi.accendi(keys.H1)
        if data['method'] == 'spegniH1':
                metodi.spegni(keys.H1)

        if data['method'] == 'accendiH3':
                metodi.accendi(keys.H3)
        if data['method'] == 'spegniH3':
                metodi.spegni(keys.H3)

        if data['method'] == 'accendiH2':
                metodi.accendi(keys.H2)
        if data['method'] == 'spegniH2':
                metodi.spegni(keys.H2)

        if data['method'] == 'accendiH4':
                metodi.accendi(keys.H4)
        if data['method'] == 'spegniH4':
                metodi.spegni(keys.H4)

        if data['method'] == 'accendiH6':
                metodi.accendi(keys.H6)
        if data['method'] == 'spegniH6':
                metodi.spegni(keys.H6)

        if data['method'] == 'accendiH4':
                metodi.accendi(keys.H4)
        if data['method'] == 'spegniH4':
                metodi.spegni(keys.H4)

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.username_pw_set(keys.TOKEN_PW_SET)
client.connect(keys.THINGSBOARD_HOST, keys.THINGSBOARD_MQTT_PORT, keys.KEEP_ALIVE)

try:
    client.loop_forever()
except KeyboardInterrupt:
    client.disconnect()
```

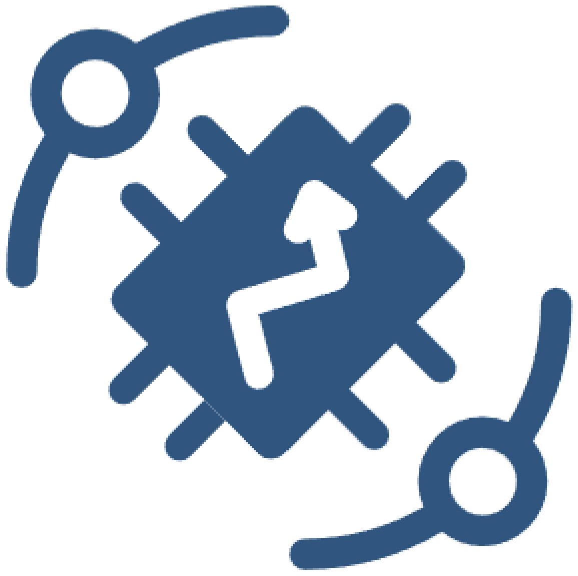
Data capture in the ThingsBoard

Activating the sensors for data acquisition and storing on ThingsBoard

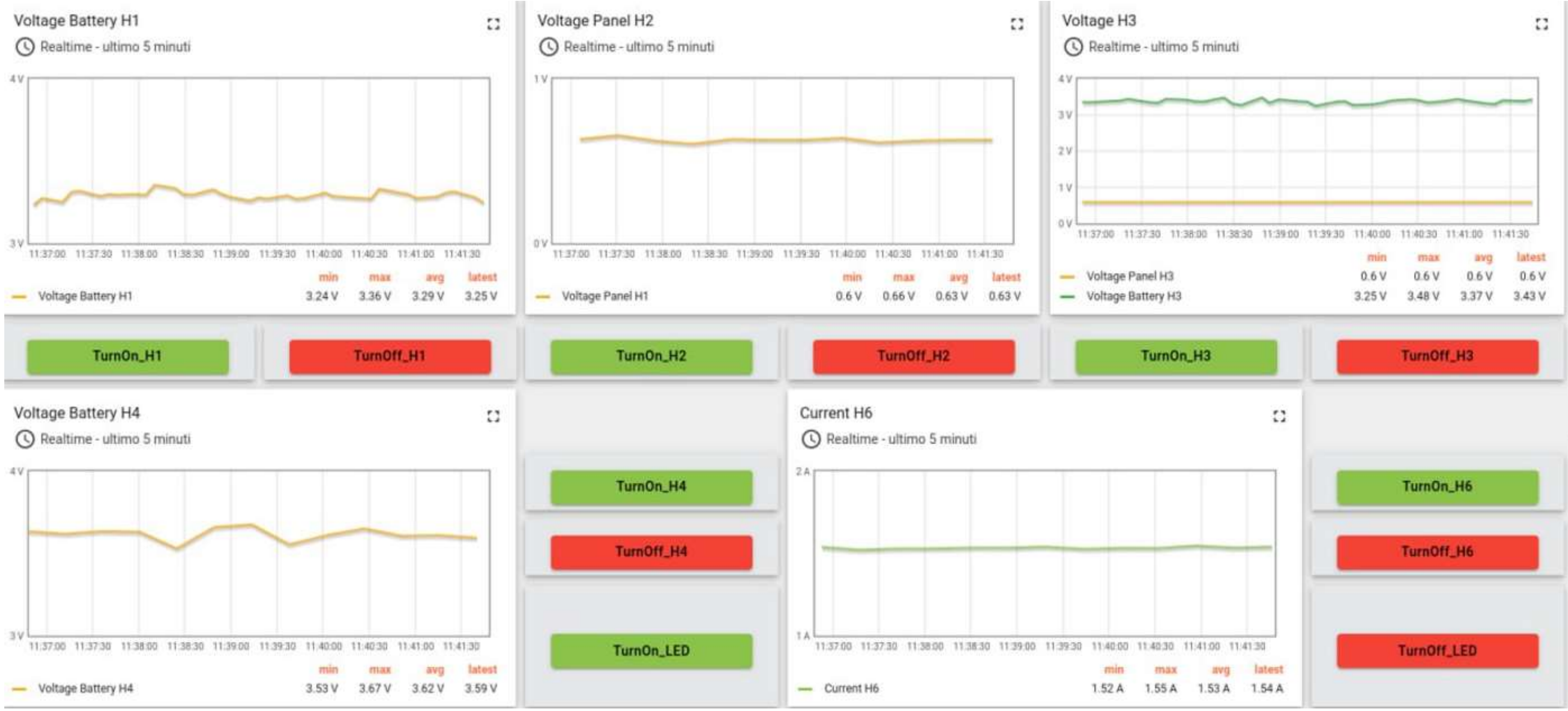
ThingsBoard is an IoT platform for real-time capture, storage, and display of sensor data

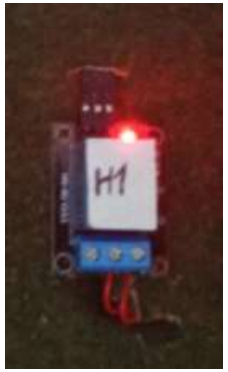
The platform supports a variety of communication protocols, including MQTT, CoAP, and HTTP, making it suitable for data acquisition from various types of sensors

ThingsBoard allows the creation of custom dashboards, which can be used to display sensor data in a more intuitive and understandable manner



Data capture in the ThingsBoard

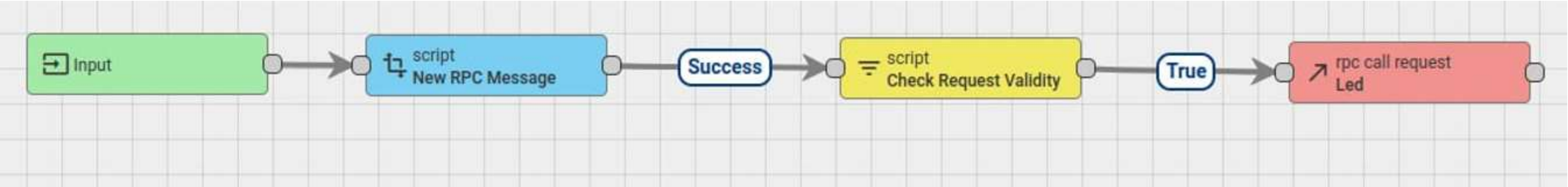




Turn on

Turn off

Rule Chain



Input
script
New RPC Message
Success
script
Check Request Validity
True
rpc call request
Led

Conclusion

The simulation illustrates how to create an example of a Smart Grid with sensors and microcontrollers

The visualization of the generated data also adds value to the control of the grid

In the future, there may be a task of leveraging artificial intelligence to utilize weather data for predicting the performance of wind turbines and photovoltaic systems